FERMILAB-Conf-88/167-T
October, 1988

Status of the Fermilab Lattice Supercomputer Project*

P. Mackenzie, E. Eichten, G. Hockney, H. B. Thacker, and D. Toussaint[†]
Theoretical Physics Group
Fermi National Accelerator Laboratory
Batavia, IL 60510 USA

R. Atac, A. Cook, J. Deppe, M. Fischler, I. Gaines,
D. Husby, T. Nash, T. Pham, and T. Zmuda
Advanced Computer Program
Fermi National Accelerator Laboratory
Batavia, IL 60510 USA

Abstract

Fermilab has completed construction of a sixteen node (320 megaflop peak speed) parallel computer for lattice gauge theory calculations. The architecture was designed to provide the highest possible cost effectiveness while maintaining a high level of programmability and constraining as little as possible the types of lattice problems which can be done on it. The machine is programmed in C. It is a prototype for a 256 node (5 gigaflop peak speed) computer which will be assembled this winter.

1 Introduction

The Advanced Computer Program Multi-Array Processor System (ACPMAPS) [1] is a massively parallel floating point computer designed for lattice gauge theory and other grid-oriented problems. In the last few years, lattice gauge theorists have seen the achievement of two important milestones on the road to first-principles calculations of hadronic properties with reliable estimates of the accuracy of the calculations. First, apparently reliable Monte Carlo calculations of simple quantities in pure gauge theory (without quarks) have begun to appear, starting with the

*Talk given by Paul Mackenzie at DPF 88, Storrs, Connecticut, Aug. 15-18, 1988.
†Permanent address: Dept. of Physics, University of Arizona

temperature of the deconfining transition in pure gauge theory. In addition to providing confidence that the program of large scale Monte Carlo calculation in four dimensional nonabelian gauge theory is succeeding, these calculations have put estimates of the computing needs for full QCD calculations on a firmer footing. Lattice sizes of $32^4 - 64^4$, requiring 1 - 20 Gigabytes of data memory seem to be a reasonable guess.

Second, the search for improved algorithms for the most difficult part of QCD calculations, the inclusion of the effects of the sea quarks in hadron calculations, has been very successful.[2] On large lattices, the hybrid Monte Carlo algorithms which are now coming into use are roughly 10^4 times faster than the seven year old algorithms from which they descend, and are now "only" about 100 times slower than the analogous algorithms for pure gauge theory. Further improvements using nonlocal techniques such as Fourier acceleration and multigrid methods are quite possible.

The generation of special purpose computing machines now coming on line for lattice gauge theory are around 10^4 times as powerful as the VAXes on which the first Monte Carlo calculations of the hadron spectrum were performed in 1981.[3] The combined improvement in hardware and algorithmic calculational power of 10^8 since 1981 approaches but does not quite meet the demands of full QCD if the more conservative estimates of calculational needs are correct. The remaining factor must and almost certainly will be met by further improvements in hardware and algorithms.

The Fermilab lattice machine was designed to provide very large amounts of computational power at reasonable cost, without compromising the programmability required for further rapid algorithm development. A sixteen node machine (320 megaflops, peak speed) has been constructed and is being tested. It is a prototype for a 256 node machine (5 gigaflops, peak speed, 2 gigabytes of data memory). A more detailed discussion of the hardware and the performance of the machine will appear in [4].

2 Architecture

The architecture shares some features with all lattice machines which have been built since the first Columbia machine[3]: it is based on a massively parallel set of nodes each containing fast floating point hardware and a lot of data memory. Each node typically performs calculations for the subset of lattice sites whose fields are stored in its local memory.

Among the most important differences between our machine and others of its type are programmability in an ordinary high level language, and the operation of the individual nodes totally asynchronously both in computation and in communication. These resulted from two fundamental design goals for the machine: that it be programmable in a high level language and that the architecture of the machine constrain as little as possible the types of lattice problems which can be done on it. Programmability in C and Fortran (we are currently using C) was made possible by the Weitek XL chip set. This chip set which contains a 20 megaflop (peak speed) floating point unit, an integer processor, and an instruction sequencer, is programmable as a whole using compilers supplied by Weitek. Each node contains one chip set, eight megabytes of data memory and two megabytes of code memory. The memory is 100 ns. page mode DRAM.

The desire to have the architecture constrain possible physics problems as little as possible led to the use of totally asynchronous operation of the nodes (MIMD) and completely transparent nonlocal communication between nodes. MIMD architecture is very flexible: it can handle problems which are awkward or impossible for single instruction, multiple data (SIMD) architectures, such as heat bath and incomplete LU decomposition algorithms and random lattice problems. The allowed sizes and shapes of the lattices are independent of the details of the hardware. The node structure of the machine can be made invisible in the high level code, resulting in improved programmability. MIMD has the potential for communications bottlenecks which cannot occur with SIMD, but these do not seem to be very severe in the codes we have tested so far.

Asynchronous internode communication is made possible by a network of switch crates into which the nodes are plugged. They handle full sixteen port crossbar switching at bandwidths of 20 megabytes/second per channel. This yields a total bandwidth of 2.56 gigabytes/second for a 256 node machine. The crates allow any node to access the memory of any other node without knowing where the other node is located on the network. With the current switch crate hardware, systems of up to 2048 nodes are possible before this transparent nonlocal communications feature is lost.

3 Software and Programming

The main ("control") program which controls global tasks such as defining lattices and fields and starting global operations on fields executes on one of the nodes (the "control" node) which is identical to all of the other nodes except in software. Global (lattice-wide) objects and operations such as global fields and operations on them, which might be implemented as arrays and for-loops on a one-CPU machine must be handled with extensions to the language when the data and operations are spread over many CPUs. These extensions should reflect as much as possible the concepts of the problems to be solved, so we consider some of the fundamental concepts of lattice problems:

Objects	"Algebras"	Operations
sites s directions d paths p	grid:	$s' = s + d$
unitary matrix u quark q	SU(3):	$u_3 = u_1 u_2$
fields U, Q	Dirac equation:	$(\not{D} - m)Q = \delta$

In object oriented languages such as C++ which are just beginning to appear, these concepts could be added to the language as new data types and operations. To us, the structure rather than the syntax is most important and we implement them with C's typedef facility and C subroutines called from the control program. Our software package for doing grid-oriented problems on parallel machines is called CANOPY.

Consider, for example, the following set of statements from a control program.

```
lat1 = periodic_grid( NDIM, latsize);
q  = site_field( lat1, sizeof(quark) );
q1 = site_field( lat1, sizeof(quark) );
complete_definitions();
```

The function **periodic_grid()** tells the system that our calculation will be done on a lattice of **NDIM** dimensions whose sizes are contained in the array **latsize**, and which will be identified by **lat1**. The function **site_field()** tells the system that memory will be required for two fields identified by **q** and **q1**, each with **sizeof(quark)** bytes for each site of **lat1**. The function **complete_definitions()** calls routines which assign specific sites to specific nodes, allocate memory in the nodes for the field data and site structures, and set up structures for each site pointing to the memory areas of adjacent sites of the lattice.

The loop over lattice sites in a function which operates on a field q with an operator **dslash** and stores the result in another field **q1** is replaced by the statement

```
do_task(dslash_, lat1,
        PASS,  q, sizeof(q),
        PASS, q1, sizeof(q1),
        END);
```

The system function **do_task()** passes to all the nodes a pointer to the user supplied function **dslash_** and an identifier of a list of sites on which to operate, which may be the entire lattice **lat1** or some previously defined list of sites such as **red_sites**. A system subroutine on the node, invisible to the user, calls **dslash_** for the sites in the set of sites which have been assigned to the node. **Do_task** may be used to pass (**PASS**) to the nodes arguments required by the function (such as the field identifiers **q** and **q1**), and to integrate (**INTEGRATE**) data returned from the individual nodes. The site subroutines access and replace data from global fields with system functions such as

```
pq = field_pointer( q, &site1 );
```

They determine whether the desired data is already present in the node's local memory and open a channel to the communications hardware if necessary.

CANOPY is written in C and is easily portable to any single-CPU or MIMD multiprocessing system with support for UNIX calls. Thus, programs can be tried out on small lattices on a workstation, and migrate to the production machine without changing any code. To date, the software has been ported to the ACPMAPS system, and to an ULTRIX MicroVAX, a MIPS M500 system, a Sun workstation and an IBM PC running Turbo C. An important consequence of using CANOPY is that the style of coding is guided into being structured and modular. The benefits of this range from more readable code, through easier code modification and debugging, to the ability to confidently optimize critical sections of the code. The overhead associated with CANOPY ranges from 0 to 15%.

4 Performance

At the present time, only one algorithm has been carefully optimized, a pseudo-heat bath code for generating gauge configurations. The code, and all of the CANOPY library, were first written in C. This yielded a link update time of 2.1 ms/link/node. The compiled code of key routines such as the transcendental functions, SU(3) multiplication, and key CANOPY routines was replaced with hand optimized code in a modular way as the time consuming parts of the code were identified. The current link update time per node for this code is 0.60 msec on a single node and .66 msec on the full 16 node machine. Link update times for this algorithm have been published for two of the other two QCD machines. [5,6] Their performance is roughly 40% slower than this if normalized to the peak speeds of the machines. The few tens of per cent in relative efficiency, while very encouraging, is not the only important point, and will vary from algorithm to algorithm. The most important point is that a very high efficiency was obtained using high level programming and modular optimization.

5 Current Status

The sixteen node prototype machine is finished and is being tested. Physics calculations will begin as soon as testing is complete. Most of the parts for the 256 node machine have been purchased, at a cost of a little over one million dollars. Assembly of the 256 node machine will begin this winter.

References

[1] P. Mackenzie et al., in Field Theory on the Lattice, ed. A. Billoire et al., Nuc. Phys. B (Proc. Suppl.) 4, 580 (1988); T. Nash et al., talk given at the Adriatico

Conference on the Impact of Digital Microelectronics and Microprocessors on Particle Physics, International Centre for Theoretical Physics, Trieste, Italy, March 28-30, 1988.

[2] Don Weingarten, to appear in the Proceedings of the 1988 Symposium on Lattice Field Theory, Fermilab, Sept. 22-25, 1988, to be published in Nuc. Phys. B.

[3] For a review of special purpose QCD machines, see Norman Christ, to appear in the Proceedings of the 1988 Symposium on Lattice Field Theory, Fermilab, Sept. 22-25, 1988, to be published in Nuc. Phys. B.

[4] Mark Fischler and George Hockney et al., to appear in the Proceedings of the 1988 Symposium on Lattice Field Theory, Fermilab, Sept. 22-25, 1988, to be published in Nuc. Phys. B; D. Husby et al., to appear in the proceedings of the 1988 Nuclear Science Symposium, Orlando, Florida, R. Atac et al., to appear in the proceedings of the 1988 Nuclear Science Symposium, Orlando, Florida.

[5] Enzo Marinari, in Field Theory on the Lattice, ed. A. Billoire et al., Nuc. Phys. B (Proc. Suppl.) 4, 3 (1988).

[6] Norman Christ, in Field Theory on the Lattice, ed. A. Billoire et al., Nuc. Phys. B (Proc. Suppl.) 4, 241 (1988).